IN THE CLAIMS

Noting that a clean listing of claims as amended is appended hereto, please amend the claims as indicated in the rewritten claims listed below:

Claim Amendments:

1. (Canceled)

2. (Currently amended) A method for verifying computer instructions in ~~In~~ a computer that includes at least one processor that executes instructions stored in a memory, ~~which is~~ the memory being organized into separately addressable memory blocks, ~~a method for verifying the validity of instructions~~ the method comprising:

identifying a next instruction to be executed when executing a series of instructions;

for ~~at least one current instruction that has been identified~~ the next instruction, ~~for submission to the processor for execution,~~ determining an identifying value for a ~~current~~ memory block that contains the ~~current~~ next instruction;

determining whether the identifying value satisfies a validation condition, wherein the determining as to whether the identifying value satisfies the validation condition requires comparing the identifying value of the ~~current~~ memory block with a set of reference values;

allowing execution of the next instruction by the processor when ~~if~~ the identifying value satisfies [[a]] the validation condition~~, allowing execution of the current instruction by the processor~~; and

generating a response when ~~if~~ the identifying value does not satisfy the validation condition~~, generating a response~~;

whereby the ~~current~~ next instruction is verified dynamically before being executed.

3. (Currently amended) The method of ~~A method as in~~ claim 2, ~~further comprising: including in the set of reference values at least one validation entry corresponding to at least one identifying value for predetermined contents of a known, valid memory block; in which~~ wherein the validation condition is that the identifying value of the ~~current~~ memory block matches any ~~validation entry~~ reference value in the set of reference values.

4. (Currently amended) The method of ~~A method as in~~ claim 2, ~~further comprising: including in the set of reference values at least one invalidation entry corresponding to at least one identifying value for predetermined contents of a known invalid memory block; in which~~ wherein the validation condition is that the identifying value of the ~~current instruction~~ memory block differs from ~~all invalidation entries~~ each reference value in the set of reference values.

5. (Canceled)

6. (Currently amended) A method for verifying computer instructions in a computer that includes at least one processor that executes instructions stored in memory, the memory being organized into separately addressable memory blocks, the method comprising:

for at least one current instruction that has been identified for submission to the processor for execution, computing a hash value as a function of a sub-set of contents of a current memory block that contains the current instruction;

determining whether the hash value satisfies a validation condition by comparing the hash value of the current memory block with a set of reference values;

if the hash value satisfies the validation condition, allowing execution of the current instruction by the processor;

if the hash value does not satisfy the validation condition, generating a response;

wherein the computing of the hash value comprises applying a mask to the current memory block, the mask being a data structure that designates at least one byte of the current

memory block to be ignored in the computing of the hash value, the data structure designating less than an entire memory block so that the hash value[[s]] is based on only part of the contents of ~~both~~ the current ~~and the reference~~ memory block[[s]].

7. (Currently amended) The method of ~~A method as in~~ claim 6, further comprising:

identifying, ~~in at least one reference~~ potentially non-constant contents of the current memory block, the non-constant contents being ~~non-indicative contents that are~~ valid but ~~that are at least potentially non-constant such~~ changeable so that they do not indicate validity of the ~~reference~~ current memory block as a whole; and

configuring the mask so that the mask designates at least the non-constant contents to be ignored when generating the hash value~~; before or when computing the hash value for the reference memory block, applying a mask to the contents of the reference memory block such that the non-indicative contents do not influence the computed hash value; and~~

~~before or when computing the hash value for the current memory block, applying the mask to the current memory block contents~~.

8. (Currently amended) The method of ~~A method as in~~ claim 2, further comprising:

for each of ~~a plurality of~~ the separately addressable memory blocks, indicating in a structure whether ~~each respective~~ the memory block is valid ~~validated~~; ~~and~~

accessing the structure to determine whether the memory block is valid prior to the determining of the identifying value; and

performing the determining of the identifying value when the structure does not indicate that the memory block is valid and ~~for each current instruction from a memory block whose structure indication is that it is validated,~~ directly allowing execution of the ~~current~~ next instruction when the structure indicates that the memory block is valid.

9. (Currently amended) The method of ~~A method as in~~ claim 8, wherein the structure comprises a group of hardware attribute indicators, and wherein ~~in which~~ the ~~step of~~ indicating in the structure whether the plurality of memory blocks is validated comprises ~~causing a corresponding entry to be made in a group of~~ setting one of the hardware attribute indicators, the one hardware attribute indicator corresponding to the memory block.

10. (Original) A method as in claim 9, in which the hardware attribute indicators are execute and write permission attributes associated with an entry in a translation lookaside buffer.

11. (Currently amended) The method of ~~A method as in~~ claim 8, wherein the structure comprises a software data structure, and wherein ~~in which~~ the ~~step of~~ indicating in the structure whether the plurality of memory blocks is validated comprises making a corresponding entry in the [[a]] software data structure.

12. (Currently amended) The method of ~~A method as in~~ claim 8, ~~further comprising: performing~~ the ~~steps of~~ determining of the identifying value for the ~~current~~ memory block and the determining of whether the ~~comparing the~~ identifying value ~~of the current memory block with a set of reference values~~ satisfies the validation condition are performed only when the structure does not indicate that the memory block is valid ~~for current instructions located in memory blocks not indicated in the structure as being validated; and~~, the method further comprising:

modifying the structure so that ~~if the identifying value of a current~~ the memory block is not indicated as being ~~validated~~ valid when the identifying value satisfies the validation condition~~, then setting the corresponding structure indication to indicate that it is validated~~.

13. (Currently amended) The method of ~~A method as in~~ claim 12, further comprising:

sensing modification of ~~any memory block~~ one of the memory blocks that the structure indicates is valid ~~for which the structure includes an indication~~ and, ~~upon sensing modification of~~

~~any such memory block~~ in response to the modification, setting its indication in the structure to indicate that the memory block is not ~~validated~~ valid.

14. (Currently amended) The method of ~~A method as in~~ claim 8, further comprising:

determining a branch history for the ~~current~~ next instruction; and

checking whether the memory blocks in which instructions in the branch history are located are ~~validated~~ valid, the validation condition including the requirement that each checked memory block in the branch history is ~~validated~~ valid.

15. (Currently amended) The method of ~~A method as in~~ claim 2, ~~further comprising performing the steps of~~ wherein the determining of the identifying value[[s]] ~~for current memory blocks, comparing the identifying values with the set of reference values,~~ and the determining as to whether the validation condition has been satisfied are performed only after ~~the occurrence of~~ a triggering event occurs.

16. (Currently amended) The method of ~~A method as in~~ claim 15, ~~in which~~ wherein the triggering event is ~~the~~ writing of at least one new unit of code or data to any physical component within the computer.

17. (Currently amended) The method of ~~A method as in~~ claim 15, in which the triggering event is ~~the~~ an attempted execution of any instruction located on any unverified memory block.

18. (Currently amended) The method of ~~A method as in~~ claim 15, ~~in which~~ wherein the triggering event is ~~the~~ an attempted execution of any instruction located on any unverified memory block of newly installed software.

19. (Currently amended) The method of ~~A method as in~~ claim 15, further comprising triggering ~~dynamic~~ the verification of the computer instructions depending on ~~the~~ an identity of ~~the~~ a user of the computer, the user having ~~who has~~ caused ~~submission of~~ the ~~current~~ next instruction to be identified for execution.

20. (Canceled)

21. (Currently amended) The method of claim 15, further comprising triggering dynamic verification depending on a context in which the next instruction is submitted for execution, wherein ~~A method as in claim 20, in which~~ the context is a level of security clearance associated with the computer, a user of the computer, or a program of which the ~~current~~ next instruction is a part.

22. (Currently amended) The method of ~~A method as in~~ claim 2, ~~further comprising verifying~~ wherein the identifying of the next instruction is performed for only a sample of the ~~current~~ series of instructions.

23. (Currently amended) The method of ~~A method as in~~ claim 22, ~~in which~~ wherein the sample is a time-sampled sub-set of ~~current~~ the series of instructions.

24. (Currently amended) The method of ~~A method as in~~ claim 22, ~~in which~~ wherein the sample is a sequentially sampled sub-set of ~~current~~ the series of instructions.

25. (Currently amended) The method of ~~A method as in~~ claim 22, ~~in which~~ wherein the sample is a sub-set of ~~current~~ the series of instructions sampled spatially, the sampling being over a range of ~~addresses or equivalent~~ memory block identifiers.

26. (Currently amended) The method of ~~A method as in~~ claim 2, ~~in which the step of generating~~ wherein the response comprises ~~terminating~~ termination of a software entity with which the current memory block is associated.

27. (Currently amended) The method of ~~A method as in~~ claim 2, ~~in which the step of generating~~ wherein the response comprises ~~suspending~~ suspension of execution of a software entity with which the current memory block is associated.

28. (Currently amended) The method of ~~A method as in~~ claim 2, ~~in which the step of generating~~ wherein the response comprises ~~posting~~ a message posted to a user, system administrator, or other predetermined recipient.

29. (Currently amended) The method of claim 2, wherein:

the ~~In a~~ computer ~~that~~ includes a virtual machine running in a direct execution mode on an underlying hardware platform via an intermediate software layer[[,]]; and

~~the method as in claim 2, in which the step of generating~~ the response comprises a switching of an ~~the~~ execution mode of the virtual machine from the direct execution mode to a binary translation mode.

30. (Currently amended) The method of claim 2, wherein:

the ~~In a~~ computer ~~that~~ includes a virtual machine running on an underlying hardware platform via an intermediate software layer[[,]]; and

~~the method as in claim 2, in which the step of generating~~ the response includes checkpointing the state of the virtual machine.

31. (Currently amended) The method of ~~A method as in~~ claim 2, wherein the response is a first possible response, the method further comprising:

associating the first possible response with the memory block;

associating a second possible ~~different~~ response[[s]] with ~~at least two~~ a different memory block[[s]];

upon detection of failure of the ~~current~~ next instruction to satisfy the validation condition, identifying which one of the possible responses is associated with the memory block, and generating the one possible response associated with the memory block in which the ~~current~~ next instruction is located.

32. (Currently amended) The method of ~~A method as in~~ claim 2, further comprising:

associating reference values from the set of reference values with respective programs such that each association signifies that the reference value corresponds to a memory block storing instructions for one of the programs; and

tracking which of the respective programs ~~are~~ is being executed ~~within the computer by associating~~ based on which of the reference values ~~with respective predetermined programs, a~~ matches ~~between~~ the identifying value ~~of the current memory block with any validation entry in the set of reference values indicating execution of the corresponding~~ and the association between the matching reference value and the corresponding one of the ~~predetermined~~ programs.

33. (Currently amended) The method of claim 2, wherein:

the ~~In a~~ computer ~~that~~ includes a virtual machine (VM) running on an underlying hardware platform via an intermediate software layer is operable to switch the virtual machine between a direct execution mode and a binary translation mode~~, the method as in claim 2,~~; and

the series of instructions comprise ~~further comprising verifying the validity of~~ VM-issued instructions issued in conjunction with binary translation of any of the VM-issued instructions, the VM-issued instructions thereby being verified.

34. (Canceled)

35. (Currently amended) A tangible medium embodying executable code executable for dynamically verifying a computer instructions being executed by a processor of a computer, the executable code being a verification engine causing the computer to perform a method having operations of: ~~A system for verifying the validity of executable code in a computer comprising:~~

~~at least one processor;~~

~~a mechanism for identifying at least one current instruction that has been identified for submission to the processor for execution;~~

~~a memory that is organized into separately addressable memory blocks, the~~

~~a verification engine comprising computer-executable code~~

identifying a next instruction to be executed when executing the series of instructions;

for ~~at least one current instruction that has been identified~~ the next instruction, ~~for submission to the processor for execution, for~~ determining an identifying value for a ~~current~~ memory block that contains the ~~current~~ next instruction;

determining whether the identifying value satisfies a validation condition, wherein the determining as to whether the identifying value satisfies the validation condition requires ~~for~~ comparing the identifying value of the ~~current~~ memory block with a set of reference values;

allowing execution of the next instruction by the processor when ~~if~~ the identifying value satisfies [[a]] the validation condition~~, for allowing execution of the current instruction by the processor~~; and

generating a response when ~~if~~ the identifying value does not satisfy the validation condition~~, for generating a response~~;

whereby the ~~current~~ next instruction is verified dynamically before being executed.

36. (Currently amended) The tangible medium of ~~A system as in~~ claim 35, ~~further comprising: at least one validation entry included in the set of reference values corresponding to at least one identifying value for predetermined contents of a known, valid memory block; in which~~ wherein the validation condition is that the identifying value of the ~~current~~ memory block matches any ~~validation entry~~ reference value in the set of reference values.

37. (Currently amended) The tangible medium of ~~A system as in~~ claim 35, ~~further comprising: at least one invalidation entry included in the set of reference values corresponding to at least one identifying value for predetermined contents of a known invalid memory block; in which~~ wherein the validation condition is that the identifying value of the ~~current instruction~~ memory block differs from ~~all invalidation entries~~ each reference value in the set of reference values.

38. (Currently amended) The tangible medium of ~~A system as in~~ claim 35, ~~further including~~ wherein: the identifying value is a hash value that is computed as a function of contents of the memory block ~~a hashing module within the verification engine comprising computer-executable code for determining the identifying value of the current memory block by computing a hash value as a function of at least a sub-set of the contents of the current memory block; and for computing each reference value as a hash value of at least a sub-set of a known, reference memory block~~.

39. (Currently amended) The tangible medium of ~~A system as in~~ claim 38, ~~further comprising a~~ wherein some of the contents of the memory block are ignored when computing the hash value, the ignored portion being defined by a sub-set selection structure ~~for selecting only a sub-set of the current memory block for computation of the respective hash value~~.

40. (Currently amended) The tangible medium of ~~A system as in~~ claim 39, ~~in which~~ wherein the sub-set selection structure is a mask.

41. (Currently amended) The tangible medium of ~~A system as in~~ claim 35, wherein the memory block is one of a plurality of separately addressable memory blocks and the method further ~~comprising~~ comprises:

~~a structure containing an indication,~~ for each of ~~a plurality of~~ the separately addressable memory blocks, indicating in a structure [[of]] whether ~~each respective~~ the memory block is valid;

accessing the structure to determine whether the memory block is valid prior to the determining of the identifying value; and

~~validated, the verification engine being further provided with computer-executable code for~~ performing the determining of the identifying value when the structure does not indicate that the memory block is valid and directly allowing execution of the ~~current~~ next instruction when the structure indicates that the memory block is valid ~~for each current instruction from a memory block whose structure indication is that it is validated~~.

42. (Currently amended) The tangible medium of ~~A system as in~~ claim 41, wherein the structure comprises a group of hardware attribute indicators, and wherein ~~in which~~ the indicating in the structure whether the plurality of memory blocks is valid comprises setting one of the ~~containing the indications is a group of~~ hardware attribute indicators corresponding to the memory block.

43. (Currently amended) The tangible medium of ~~A system as in~~ claim 42, in which the hardware attribute indicators are execute and write permission attributes associated with an entry in a translation lookaside buffer.

44. (Currently amended) The tangible medium of ~~A system as in~~ claim 35, wherein the structure comprises a software data structure, and wherein further comprising a software

module comprising computer-executable instructions for selecting for verification only a sample of the ~~current~~ next instructions.

45. (Canceled)

46. (Currently amended) (Currently amended) The tangible medium of claim 35, wherein:

~~A system as in claim 45, in which~~ the verification engine [[is]] resides in an intermediate virtualization layer between a virtual machine and a hardware platform of the computer;

the next instruction is issued by the virtual machine; and

the ~~further provided for~~ verifying of the validity of ~~VM-issued instructions~~ the next instruction is performed while copying or translating in conjunction with binary translation of instructions for the virtual machine ~~any of the VM-issued instructions~~.

47. (Currently amended) The tangible medium of claim 46, wherein ~~A system as in claim 45, in which the verification engine is further provided for triggering~~ the response comprises switching the intermediate virtualization layer ~~to switch execution of the virtual machine~~ to ~~the~~ a binary translation mode ~~as the response~~.